EXHIBIT G

08-11-00

A/PROV

PATENT
AWR-047

Honorable Commissioner of Patents and Trademarks
Washington, DC 20231

NEW PROVISIONAL APPLICATION TRANSMITTAL LETTER

Sir:

Transmitted herewith for filing is the Provisional Patent Application of Inventor(s):

	Murat Belge
Residence:	36 Dartmouth Street Pleasant PL #1004 Malden, MA 02148
Citizenship	Turkey
Post Office Address:	Same as above
	Michael A. Tzannes
Residence:	17 Carley Road Lexington, MA 02173
Citizenship	United States
Post Office Address:	Same as above
	Halil Padir
Residence:	85 Carlton Lane No. Andover, MA 01845
Citizenship	United States
Post Office Address:	Same as above

PATENTS
AWR-047

For: Characterization of Transmission Lines Using Broadband Signals In A Multi-Carrier DSL System

Enclosed are the following papers required to obtain a filing date under 37 C.F.R. §1.53(c):

—	Sheets of Informal Drawings
33	Pages of Specification, Drawings & Tables
—	Claims

The following papers, if indicated by an ☒, are also enclosed:

- ☐ A Declaration and Power of Attorney
- ☐ An Assignment of the invention
- ☐ An Information-Disclosure Statement, Form PTO-1449 and a copy of each cited reference
- ☐ A Small-Entity Declaration
- ☒ A Certificate of Express Mailing, Express Mail Label No. EE437022773US

Basic Fee: $150

☒ A check in the amount of $150 is enclosed to cover the Filing Fee.

Please address all communications and telephone calls to the undersigned.

Respectfully submitted,

Gail Leonick

Gail Leonick
Aware, Inc.
40 Middlesex Turnpike
Bedfored mass. 01730

PATENT
AWR-047

UNITED STATES PROVISIONAL PATENT APPLICATION

of

Murat Belge
Michael A. Tzannes
and
Halil Padir

for a

Characterization of Transmission Lines Using Broadband Signals In A Multi-Carrier DSL System

References Cited

[1] Provisional patent by D. Krinsky and R. Pizzano, "*Multicarrier Modulation System with Remote Transmission Mode*", serial no: 60/174,865, date filed: 1/7/2000.

[2] Provisional patent by M. Belge, "*Estimation of the Loop Length and Bridged Tap Length of a Transmission Line*", serial no: 60/174,866, date filed: 1/7/2000.

1 Background of the Invention

Rapid developments in the computer industry and the availability of affordable hardware created the internet where any user having a communication link between his/her home and the computers in centralized locations can access publicly available information. Users of the internet are connected to the communication network through a link that includes a telephone line from the customer premises (CPE) to a telephone company central office (CO). A computer user requesting data transfer from an internet server is faced with the limited bandwith of the connection between the his/her home and the central office. As more and more information is being created and stored in digital format, the demand from users to access large data files is increasing making it crucial to find new and faster ways of transferring data. One way of achieving faster data transmission is to increase the bandwith of the transmission line between users and the CO by replacing the current metallic conductors with fiber or using better quality metallic conductors with increased bandwith. But such an approach is costly and requires a substantial investment by the telephone companies.

Recent developments in digital signal processing and telecommunications have resulted in the digital subscriber line (DSL) technology enabling a high speed data link over existing twisted pair telephone lines. Alhough a couple of different DSL systems had been proposed multi-carrier systems have quickly gained popularity and been standardized. Multi-carrier DSL systems operate on the principle of frequency division multiplexing where seperate frequency bands are used to transfer data from the CPE to the CO and vice versa. The portion of the bandwith allocated for transmitting data from the user's computer to the CO is called the up-stream (US) channel and the portion of the bandwith allocated for passing data from the CO to the user's computer is called the down-stream (DS) channel. Since in a typical internet session the amount of data being transmitted from the CO to the user's computer is much larger than the amount of data transmitted from the user's computer to the CO, the bandwith allocated for the DS channel is usually much larger than the bandwith allocated for the US channel (typical ratios are 4 to 1 or 8 to 1). The bandwith allocated to the US and DS channels are partitioned into a large number of sub-bands which are sufficiently narrow so as to allow the distortions introduced by the line to be described as an attenuation and a phase shift. These parameters can be measured in a training session prior to establishing the data link by sending and receiving a predefined signal on each subband. The amount of data that can be sent in a sub-band is limited by the signal to noise ratio (SNR) in that sub-band which is the signal strength described by the line attenuation divided by the noise power. Each of the sub-bands in the multi-carrier DSL system is used to transmit data that is consistent with the SNR on that sub-band and maximum allowable error bit rate. Multi-carrier DSL system operating with the described principles are able to achieve data rates as high as ten million bits per second.

Although the multi-carrier DSL systems are promising because they offer a cost effective way of opening current telephone lines to high speed data transmission traffic, there are important problems in the installation and maintenance phases of DSL deployment that prevents rapid wide spread deployment. Existing telephone lines were initially installed for voice only transmission which can be done by using only a small bandwith. Multi-carrier DSL system require utilizing a bandwith much larger than that required by the voice transmission. At high frequencies line conditions that don't affect the voice transmission become important factors limiting the digital data transmission rate. For example, the line attenuation is related to the loop length. The strength of the signals sent from either CO or user's computer decreases with distance. Small open circuited twisted pairs, called bridged taps (BT), connected in shunt with working twisted pairs, while not effecting voice transmission, cause periodic dips in the attenuation function of the line at certain sub-bands and hence degrade the performance of the DSL service. Telephone lines are usually bundled as 25 or 50 twisted pairs in a cable. Close proximity of the twisted pairs in the cable causes the signals generated by various DSL services carried by a specific telephone line to be picked up by the remaining telephone lines in the bundle. These signals are perceived as additive noise components because they are unpredictable and meaningless for all but one of the telephone lines carrying the service. The interference entering the telephone lines through some coupling path with other telephone lines are called crosstalk. There may be other sources of noise in a telephone line which are caused by the reception of electromagnetic (EM) waves transmitted by various sources such as AM stations or electrical devices such as hair dryers, dimmer switches, alarm systems etc. The most detrimental of these EM sources are generally AM stations. Since no two telephone lines are the same and the availibility and the quality of a DSL link depend on the conditions of the line as described above, it is very important to be able to qualify telephone lines for DSL service and maintain the link once the service is established. It is a challenge to ease the installation and maintenance issues. To decrease the cost associated with service qualification and maintenance, it is preferrable to qualify and maintain telephone lines remotely without sending a technician to the customer premises.

It is the object of the present invention to provide a system for the qualification, maintenance and monitoring of telephone lines for DSL service by taking advantage of the DSL tranceivers in the CO and the CPE sites.

2 Summary of the Invention

Establishing a digital data link between the computer in user's home and the servers connected to the backbone of the central office requires DSL tranceivers handling the data transmission with the basic principles outlined in the previous section. Each of the tranceivers at either side of the link, the CO and the CPE, are called modems. The CO and the CPE modems consist of some analog harware to perform analog signal transmission and reception and a digital section which consists of a digital signal processing (DSP) chip and an application specific integrated circuit (ASIC) handling sophisticated signal processing operations. Because of the high data rate associated with the DSL service, the DSP chip must complete the necessary processing and manipulation on digital data in short time intervals. That is the DSP chips used in the CO and the CPE modems must be, by definition, very powerful. The present invention takes advantage of the vast computational capacity of the DSL modems and the presence at the two sides of the transmission line to characterize the transmission line. The DSL modems operate as a modem in their usual state but they can be put in a different mode where they can be used as test and measurement devices.

The most important issues faced during the installation and maintenance is finding out the physical structure and the conditions of the line so that the a decision can be made regarding the suitability of the loop for DSL service and necessary steps can be taken to improve the telephone line so that the service providers can offer better DSL service to their customers. For example, if a bridged tap causing a substantial data rate reduction is found the telephone company may send a technician to remove it. In general the following information is very useful for characterizing the transmission line

- the loop length
- the detection of the bridged taps and the estimation of their lengths and locations
- the detection of interferers on the line.

After the installation the link must be monitored in order to ensure service quality. This requires determining changes in the transmission environment which can be again accomplished by using the signal processing capabilities of the DSL modem.

In the present invention the CO and the CPE modems are used as test points and the test process consists of collecting specific data sets during modem training, postprocessing the data to ease the use and interpretation and finally extracting plain English results regarding the line conditions. In modem training the objective is to do measurements and determine the parameters of the transmission line so as to allow restoration of the original signals transmitted by the CPE and the CO modems. These signals are distorted by the transmission line through attenuation and phase shift and further degraded by the noise. The CO and CPE modems go through a pre-defined and standardized set of states to learn the parameters of the entire communication system. They transmit and receive signals known to each modem. These signals help characterize the transmission line but the CO and the CPE modems do not use these signals for anything other than improving the signal transmission and reception. The current invention requires adding a data collection software to either the CO or the CPE or preferably to both so that some of the data sets already used in the modem training can be collected with improved accuracy and can be saved for further analysis. The data collection software also allows some new data sets to be obtained.

Because the CO and the CPE modems operate by frequency division multiplexing principle, the data collected at the CPE and CO modems are different in the sense that the CPE modem transmits in the US channel and receives in the DS channel and the CO modem transmits in

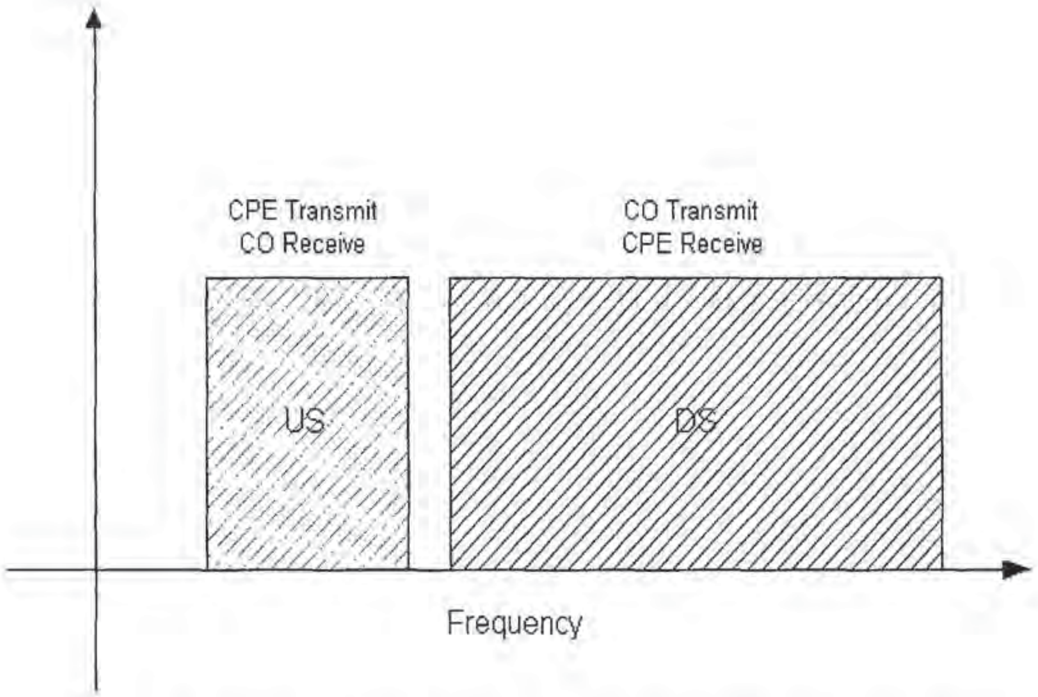

Figure 1: US and DS channel allocation for a typical ADSL multi-carrier DSL system.

the DS channel and receives in the US channel. Therefore the bandwith of the data collected at the CPE is limited to the bandwith of the DS channel and similiarly the bandwith of the data collected at the CO is limited to the bandwith of the US channel. As an example, the bandwith allocation for an ADSL system is illustrated in Figure 1. To summarize, as a result of the modem training we have the following:

- US data collected and saved in the CO modem.
- DS data collected and saved at the CPE modem.

Note that the test process as mentioned above makes use of the standard modem training and therefore relies on the existence of both the CO and the CPE modem. We call such a test process a *double-ended* test.

In a double ended test, the DS data collected at the CPE can be transferred to the CO site to be analyzed by service technicians. This requires the ability to establish a special diagnostic link between the CO and the CPE for passing the diagnostic data even if the standard DSL link fails. This can be accomplished, for example, by the method that was described in reference [1]. In the case a diagnostic link cannot be established only local data (US data at the CO test point and DS data at the CPE test point) is available for further analysis.

The telephone company may want to perform a *single-ended* test from either the CO or the CPE site to pre-qualify customer lines for DSL service. Also, a computer manufacturer who installs DSL modems into its computers may want to perform a single-ended test so that the customer can determine what kind of DSL service to order. In these cases we can continue using the signal processing cababilities of the DSL modem in a different fashion. In a double-ended test, one of the modems acts as a signal generator and the other works as a signal receiver. In the single-ended testing the same DSL modem acts both as the signal generator and signal receiver for characterizing the customer line.

The objective of the present invention is to to analyze the data collected as a result of a single or double-ended test process and characterize the transmission environment by plain English interpretation results such as loop length, bridged tap lengths, interferer types etc., that can be easily understood by unskilled technicians and at the same time producing graphs for the visual inspection of the line conditions. Specifically, the present invention embodies

1. a method for using DSL signal processing engine to characterize transmission lines.
2. a method for tracking changing line conditions.
3. a method for compensating the effects of analog front end (AFE) of the CO and the CPE modems to better characterize the transmission line.
4. a method and algorithm for estimating the loop length and detecting bridged taps and estimating their lengths from DS data (the current presentation of this algorithm is an improved version of the method given in reference [2]).
5. a method and algorithm for estimating the loop length and detecting the bridged taps and estimating their lengths from US data.
6. a method an algorithm for estimating the loop length and detecting the bridged taps and estimating the location and the lengths of the detected bridged taps using single-ended testing.
7. a method and algorithm for detecting crosstalk interferers on the line and estimating the power level of the detected disturbers.
8. a method and algorithm for detecting EM disturbers such as AM stations or amateur radio stations and estimating their frequencies and power levels.
9. a method and algorithm for estimating the data rate reduction caused by the presence of various interferers on the line.
10. a method and algorithm for pre-qualification of a telephone line where the maximum data rate that can be supported by the telephone line is estimated by single-ended testing.
11. a method and algorithm for obtaining visually displayable data from the raw data collected in the CO and CPE modems.
12. a method for displaying the results in plain English for unskilled technicians.

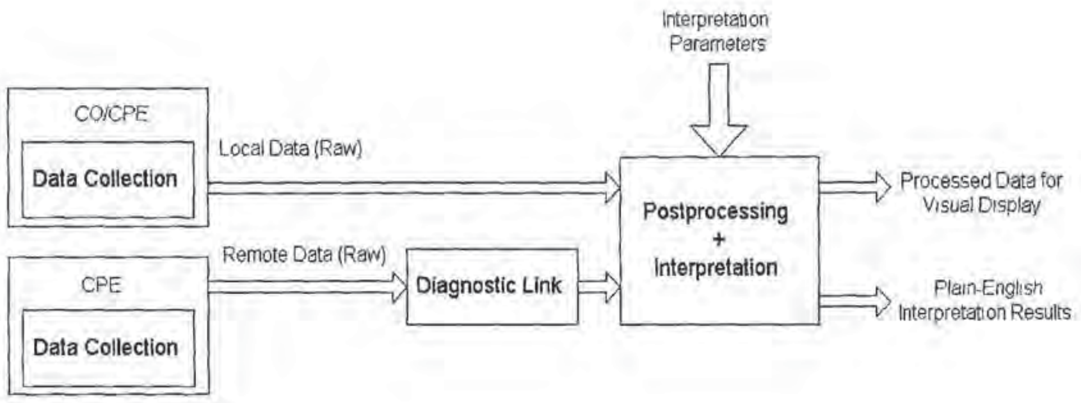
Interpretation Parameters
CO/CPE
Data Collection
Local Data (Raw)
Postprocessing + Interpretation
Processed Data for Visual Display
Remote Data (Raw)
CPE
Diagnostic Link
Plain-English Interpretation Results
Data Collection

Figure 2: Various functional blocks of the present invention.

Although the above description disscusses the use of ADSL for this application, it is clear that any form of DSL (including VDSL, SDSL, HDSL, HDSL2) could also be used. It should also be noted that the methods and algorithms listed above are applicable to both discrete multi-tone (DMT) and discrete wavelet multi-tone (DWMT) DSL systems.

3 Detailed Description of the Invention

The present invention, from a functional point of view, involves three primary steps as illustrated in Figure 2:

1. Data collection,
2. Postprocessing,
3. Interpretation.

The data collection process is always implemented in the DSL modem and responsible for producing the raw data for postprocessing and interpretation. The postprocessing part is designed to ease the use and interpretation of the raw data. Its responsibilities include calibration, filter compensation, estimation of remote SNR tables from bits and gains tables, and data rate conversion. Detailed descriptions of each of these processes and sample software are provided in Appendices in section A.

Interpretation part of the present invention is responsible for analyzing the postprocessed data and extracting plain English, easily understandable results about the line conditions.

We have a number of different interpretation routines

- **Loop Length/Bridged-Tap Length Estimation (Appendix B.1):** Estimates the loop length and tries to determine the presence of the bridged tap on the transmission line. If a bridged tap is detected, its lengh is also estimated. The algorithm operates by comparing a model of the transfer function of the line, which is parametrized in terms of the loop length and the bridged tap lengths and locations, to the actual measured transfer function of the line. We have three different algorithms to estimate the physical structure of the loop depending on which data set is being used (US, DS or single-ended time domain reflectometry data set). These three algorithms are described in detail in Appendix B.1.1 - B.1.3.

- **Interferer Detection (Appendix B.2):** Performs the task of identifying crosstalk and EM disturbers on the line by analyzing the measured power spectrum of the noise. We have two different algorithms for identifying crosstalk sources (Appendix B.2.1) and detecting EM sources (Appendix B.2.2).

- **Data Rate Reduction Estimation (Appendix B.3):** Estimates the data rate reduction caused by the presence of the disturbers on the transmission line.

- **Data Rate Estimation (Appendix B.4):** Estimates the maximum data rate that the transmission line can support by performing a single-ended test. It combines the results of single-ended TDR test and the measurement of the power spectrum of the noise on the line to estimate a rough SNR profile for US and DS channels and estimates the data rate based on these SNR tables.

Appendix

A Postprocessing

Postprocessing routines perform various simple tasks such as converting raw data from one format to the other, scaling the data, and compensating for the analog and digital filters in the receive path. Below we outline the basic postprocessing routines, their functional specifications and implementational details.

A.1 Calibration

Processes the raw data in fixed point format (ICN, Reverb, TDR) so that the data appears the same way as it would if measured by a standard test equipment. The calibration routine takes the raw data array, PGA setting used to collect the data and the gain scaling (if there is any) and outputs the calibrated data. Below we provide a C-like pseudo-code for the calibration routine. Note that calibration function in the actual implementation may slightly vary depending on the raw data being used.

First we explain the conventions used in pseudo-code implementation. Below, int16 represents a 16-bit integer number. Outputs of a function is listed in square brackets. All inputs and outputs are described in the comments section immidiately after function declaration.

Pseudo-Code:

```
[float CalData] = Calibrate( int16 RawData[], int N, float PGA, int16 GScale )
// Inputs:
// RawData[] = array containing the raw data
// N = number of elements in array RawData[]
// PGA = PGA setting used to collect data
// GScale = Scaling applied to the elements of RawData[]
//
// Output:
// CalData = Output array containing the calibrated data.
{

 for (i=0; i < N; i++)
 {
     CalData[i] = 10*log10( RawData[i] * 2^GScale ) - PGA;
 }

 return CalData;
}
```

A.2 Filter Compensation

Removes the effects of AFE filters from the measured data. The filter compensation routine takes the calibrated data and the device specific frequency domain response of the AFE filters and outputs the filter compensated data.

Pseudo-Code:

```
[complex CompData] =  FilterCompensate( complex CalData[], complex CompFilter[],
                                        int N)
// Inputs:
// CalData[] = array containing the calibrated data in dB.
// CompFilter =  device specific frequency domain filter function in dB.
// N = number of elements in array CalData[]
//
// Output:
// CompData = array containing filter compensated, calibrated data.
{

 for (i=0; i < N; i++)
 {
     CompensatedData[i] = CalibratedData[i] - Filter[i];
 }

 return CompData;
}
```

A.3 Filter Compensation for Reverb Collected in Showtime

In service monitoring, CPE and CO modems collect the reverb signal received in sync frame. Since TDQ and FDQ are normally in operation in showtime, the received reverb signal is affected by the TDQ and FDQ filters. It is possible to remove the effects of TDQ and FDQ filters by doing a trivial frequency domain deconvolution.

Pseudo-Code:

```
[float CompData] = FilterCompensate(complex CalData[], complex TDQ[],
                                    complex FDQ[], int N)
// Inputs:
// CalData = calibrated data array (in dB).
// TDQ = array containing TDQ filter coefficients.
// FDQ = array containing FDQ filter coefficients (in dB).
// N = number of elements in array CalData[].
```

```
//
// Output:
// CompData = compensated data in dB.
{

 TDQ =  FFT( TDQ, N );          //FFT of TDQ coeffs in dB.
 for (i=0; i < N; i++)
 {
     TDQ[i] = 10*log10( TDQ[i] );
 }

 // Deconvolution
 for (i=0; i < N; i++)
 {
     CompData[i] = CalData[i] - FDQ[i] - TDQ[i];
 }

 return CompData;
}
```

A.4 SNR Medley from Bits and Gains Table

In two-ended provisioning, if CO or CPE is not cpable of establishing a diagnostic link, we only have the local US or DS data. However, a crude representation of the SNR table of the far end modem can be obtained through a standard link. According to the G.dmt and G.lite specs, each modem has to send a bits and gains table to the other side which indicates the number of bits assigned to each tone and corresponding fine gain. Since bit allocation is directy related to the SNR, a reverse transformation from bits and gains table to the SNR table is possible. However, the reverse transformation is not perfect. SNR's below 10 dB and above 55 dB cannot be observed.

Pseudo-Code:

```
[float SNR] = SNRFromBits&Gains( int16 Bits[], int16 Gains[], int N)
// Input:
// Bits = Far end bitloading table.
// Gains = Far end fine gains table.
// N = number of elements in Bits[] and Gains[] arrays.
//
// Ouput:
// SNR = Estimated far end SNR table.
{

 float SNRRequired[] = {0.0,   11.31, 14.32, 19.11
                        21.31, 24.46, 27.54, 30.59,
                          33.61, 36.63, 39.65, 42.66,
                          45.67, 48.68, 51.69, 54.70};

 for (i=0; i < N; i++)
 {
     SNR[i] = SNRRequired( Bits[i] ) - Gains[i];
 }

 return SNR;
}
```

A.5 Rate Conversion

Converts the data rates in units of 32,000 kilo bits per second (Kbps) to the actual rate in Kbps.

Pseudo-Code:

```
[int Rate] =  ProcessRate( int16 RawRate )
// Input:
// RawRate = raw data rate.
// Output:
// Rate = Rate in Kbps.
{

 return RawRate * 32,000;

}
```

B Interpretation

Interpretation software is responsible for extracting plain-English results from the postprocessed data. In the following sections, we provide theoretical background on the interpretation algorithms, give computational and memory requirements and elaborate on implementational details.

B.1 Loop Characterization: Loop Length Estimation and Bridged Tap Detection/Length-Estimation

The loop characterization algorithms employ a model based approach to estimate the length of the loop and the lengths of upto two bridged taps. Specifically, our channel characterization algorithm compares the measured channel impulse response to the channel impulse response of a loop model consisting of a single-gauge wire and containing upto two bridged taps. The loop length and the bridged tap lengths are the parameters of the theoretical channel impulse response. The algorithm changes the parameters of the theoretical model and evaluates the error between the measured channel impulse response and the theoretical channel impulse response. The loop length/bridged tap lengths that minimize the error function are declared as the estimated values. The presence of a bridged tap is declared only if the bridged tap length is greater than a hundred feet.

There are two separate algorithms which perform loop characterization from downstream (DS) and upstream (US) data. During modem initialization, data collection software collects the reverb signal by averaging K consecutive frames ($K \geq 64$). The received reverb signal obtained in this way is an estimate of the impulse response of the entire channel including the front end responses of the transmitting and receiving modems. The frequency domain received reverb signal is obtained in the following way:

$$Rx(f) = \frac{1}{K} \sum_{k=1}^{K} \mathrm{FFT}_N\left(rx(n)\right) \tag{1}$$

where f is a dummy variable denoting frequency and $rx(n)$, $n = 1, \ldots, N$, is the samples of the time-domain received reverb signal within a frame with N being the number of samples contained in a single frame. The equation in (1) may contain a slight abuse of notation because in reality the frequency variable f is not continuous but rather discrete and for this reason the channel impulse response is available only at a set of discrete frequencies, called tones, which are multiples of $\Delta f = 4312.5\text{Hz}$:

$$f_i = i\Delta f, \quad i = 1, \ldots, N/2 \tag{2}$$

The reverb signal is transmitted only over a portion of the entire ADSL spectrum. The reverb signal is available at 224 (96 in G.Lite) tones from $f_{32} = 32\Delta f$ to $f_{255} = 255\Delta f$ in DS channel and at 26 tones from $f_6 = 6\Delta f$ to $f_{31} = 31\Delta f$ in US channel. The DS reverb signal is collected at CPE and US reverb signal is collected at CO. While there is no difference in the data collecttion process for US or DS reverb signal, the characteristics of these two data sets are quite different. First of all the DS reverb data contains significantly more information. There are more samples of the frequency domain reverb signal available in DS direction and these samples cover an extended range in frequency domain where the effects of bridged taps on impulse response can be easily detected. There is one crucial difference between US and DS data sets which prevents using the same interpretation algorithm for both. In the DS channel, the matching of the front and impedance to the loop impedance tends to be better than the US channel. This makes it possible to use a simplified channel model for DS channel. Unfortunately, impedance matching in the US channel is not as good as in DS channel and a more complicated channel impulse response must be used. Because of the said complications in channel modeling and the lack of sufficient data samples, the US channel characterization algorithm is limited in terms of estimation accuracy and the number of bridged taps that can be detected. Currently, channel characterization software is capable of detecting upto two bridged taps from DS data and one bridged tap from US data.

Below we give theoretical details leading to the derivation of the frequency domain channel impulse response of our model and explain channel characterization from DS and US data in detail. Both DS and US interpretation algorithms employ the same least squares minimization concept where the square of the error norm between the actual and the theoretical channel impulse responses is minimized but differ in the theoretical channel impulse response used.

B.1.1 Loop Characterization from DS Data

A two-wire loop is chracterized by its characteristic impedance, $Z_0(w) = \sqrt{\frac{R+jwL}{G+jwC}}$, and its propagation constant, $\gamma(f) = \sqrt{(R+jwL)(G+jwC)}$, where $w = 2\pi f$ and R, L, G and C are the frequency dependent constants of the loop. For a perfectly terminated loop (or for a very long loop) with length d, and two bridged taps of lengths b_1 and b_2, the transfer function of the loop, $H(d, b_1, b_2, f)$, is given by

$$H(d, b_1, b_2, f) = \frac{e^{-d\gamma(f)}}{[2+\tanh(b_1\gamma)][2+\tanh(b_2\gamma)]} \tag{3}$$

In logarithmic scale

$$\log|H(d, b_1, b_2, f)| = -d\gamma(f) - \log[2+\tanh(b_1\gamma)] - \log[2+\tanh(b_2\gamma)] \tag{4}$$

Note the linear dependence of the loop loss to the length of the cable. The actual transfer function of the loop can be measured during modem initialization as explained in section . Then we try to match the measured transfer function of the loop with that of a loop of length d with two bridged taps as given in (3). In other words we try to find d, b_1, and b_2 which minimizes the following least squares error criterion:

$$\min_{d,b_1,b_2} \sum_{i=i_f}^{i_l} |H(d, b_1, b_2, f_i) - Rx(f_i)|_2^2 \tag{5}$$

where $Rx(f_i)$ is the received reverb signal sampled at $f_i = i\Delta f$ and i_f and i_l are the first and the last tones $Rx(f_i)$ is available.

An example of the operation of the algorithm is illustrated in Figure 3. Here we display the measured received reverb signal $Rx(f)$ and the theoretical model $H(d, b_1, b_2, f)$ which was obtained by finding the model parameters d, b_1, b_2 that best match the data. The loop consisted of 26 awg 6000 ft wire with a 26 awg 1300 ft bridged tap close to CPE. Model parameters best matching the observed data were found to be $d = 6000$ ft, $b_1 = 1300$ ft and $b_2 = 0$ ft.

Memory Requirements and Computational Complexity: It is obvious from the problem statement in (5) that the interpretation algorithm basically does a search over the variables d, b_1 and b_2 and finds the ones minimizing the cost function given below:

$$E(d, b_1, b_2) = \sum_{i=i_f}^{i_l} |H(d, b_1, b_2, f_i) - Rx(f_i)|_2^2. \tag{6}$$

Unfortunately, the cost function $E(d, b_1, b_2)$ is a nonlinear function of d, b_1 and b_2 and contains many local minima. Therefore many well known optimization algorithms such as Gauss-Newton cannot be used since these algorithms cannot cope with multiple local minima and

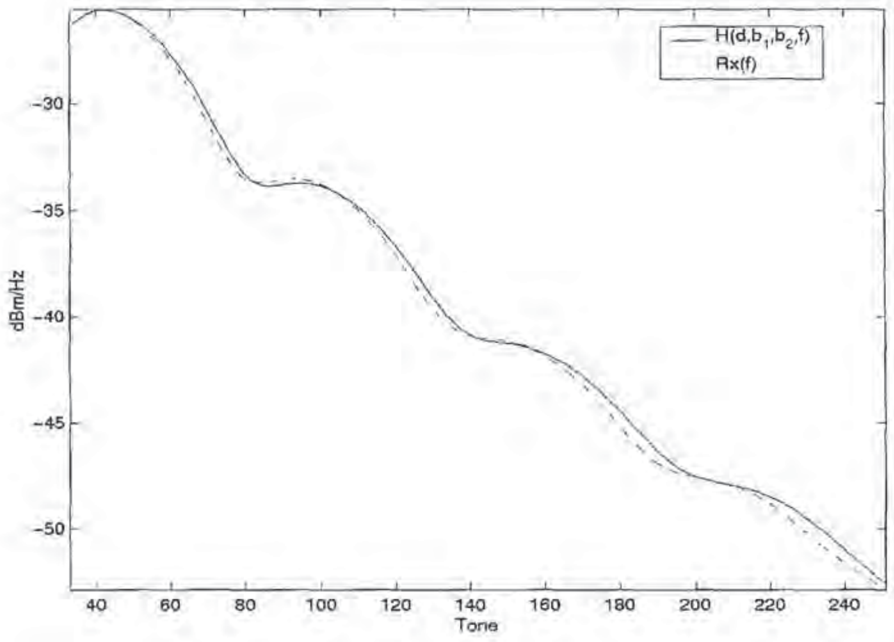

Figure 3: Observed (dashed line) received reverb signal $Rx\,(f)$ versus the theoretical channel model (solid line) $H(d, b_1, b_2, f)$ as functions of frequency for a 6000 ft loop with a single 1300 ft bridged tap.

they converge to a local minimum of the cost function. In our case we definitely want the global minimum of $E(d, b_1, b_2)$. For this reason, we decided to use a brute-force global minimization algorithm where we sample the cost function at the points (d^p, b_1^q, b_2^r), $d^p = p\Delta D$, $b_1^q = q\Delta b_1$ and $b_2^r = r\Delta b_2$ with $p = 1, \ldots, P$, $q = 1, \ldots, Q$ and $r = 1, \ldots, R$. We then choose the parameters (d^p, b_1^q, b_2^r) which results in the minimum cost among the sampled values. This requires evaluating the cost function at $P \times Q \times R$ locations.

In order to be able to compute the theoretical transfer function of the loop, $H(d, b_1, b_2, f)$, we need to store the frequency dependent propagation constant $\gamma(f)$ for a number of wires of different gauges. In the current implementation, we only use **24awg** and **26awg** wires which require $4 \times N$ locations to store the real and the imaginary parts of $\gamma(f)$ for N ADSL tones. We also need to store the AFE compensation curves which takes up N locations in memory. Depending on where we implement the algorithm, we can either compute the loop transfer function directly from (4) (as it would have been the case if the algorithm were implemented on a PC or workstation) or we may have to store $\log[2 + \tanh(b_1\gamma)]$ terms in regular intervals as required by our sampling procedure for (d^p, b_1^q, b_2^r). For example, we may pre-compute and store $\log[2 + \tanh(b_i\gamma)]$, $i = 1, 2$, from $b_1 = 100ft$ to $b_1 = 2000ft$ in 100ft intervals. Assuming we have little horsepower, we need to pre-compute and store $\log[2 + \tanh(b_1\gamma)]$ terms which takes about $20 \times N$ locations if we store the real part only. Therefore the total reqired memory is about $(20 + 4 + 1 + 3) \times N = 28 \times N$ where 2×256 locations are needed to store intermediate variables computed during the execution of the algorithm.

Although it will not be shown here, it is possible to simplify the MSE computation so that only 12 multiplications and 15 additions are needed. This means that the total computational complexity of the algorithm is about $P \times Q \times R \times (11 multiplications + 15 additions)$. This figure dominates the computational complexity. We need some additional start-up computations which is negligible compared to the above figure.

Pseudo-Code:

```
[complex H] = HmodelDS(float d, float b1, float b2, int Gauge)
// Inputs:
// d = loop length.
// b1 = bridged tap length 1.
// b2 = bridged tap length 2.
//
// Output:
// H = frequency domain loop model.
{

  float LgScale = 10.0 * log10( exp(1.0) );

  gamma = GetGamma( Gauge );      //returns frequency domain propagation function
                                  //for the wire specified by Gauge.

  //Compute freq. domain loop model.
  for (int i=0; i < N; i++)
  {
      H[i] =  - LgScale * d * gamma[i] - 10*log10( 2.0 + tanh( b1 * gamma[i] ) )
              - 10*log10( 2.0 + tanh( b2 * gamma[i] ) );
  }

  return H;
}


[float dBest, float b1Best, float b2Best] =  LoopDS( complex Rx[], int Gauge, int N)
// Inputs:
// Rx = calibrated and compensated reverb signal in frequency domain.
// Gauge = reference gauge of the wire.
// N = number of elements in Rx[].
//
// Outputs:
// dBest = Estimated loop length.
// b1Best = Estimated bridged tap length 1.
// b2Best = Estimated bridged tap length 2.
```

```
{
 int dstep = 100, b1step = 100, b2step = 100;

 float MinMSE = 1e20;
 for (d=0; d < dMax; d += dStep)
   for (b1=0; b1 < b1Max; b1 += b1Step)
     for (b2=0; b2 < b2Max; b2 += b2Step)
     {
         H = HmodelDS( d, b1, b2, Gauge );
         mse = 0.0;
         for (i=0; i < N; i++)
         {
            mse += ( H[i] - Rx[i] ) * ( H[i] - Rx[i] );
         }

         if (mse < MinMSE)
         {
            mse = MinMSE;
            dBest = d;
            b1Best = b1;
            b2Best = b2;
         }
     }
 return dBest, b1Best, b2Best;
}
```

B.1.2 Loop Characterization from US Data

Unlike the DS interpretation case, we can no longer assume that the line is terminated perfectly. The impedance mismatch at the transmitter-line connection at the CPE modem and the impedance mismatch at the receiver-line connection at the CO modem become important factors that must be taken into account. While the basic idea behind the channel characterization algorithm from the US data remains the same and involves matching a theoretical channel transfer function to the actual measured transfer function, the computation of the theoretical channel transfer function becomes much more involved. Before going into details, we point out that the channel transfer function is again measured by averaging K frames of the received reverb signal as given in (1).

Our theoretical model for channel transfer function in this case can be described in two steps. The first step consists of writing the equations for the current and voltage at the source (CPE), I_S, V_S, in terms of current and voltage at the load (CO), I_L, V_L, through the application of ABCD matrices:

$$\begin{bmatrix} V_S \\ I_S \end{bmatrix} = F^S \times A^1 \times B \times A^2 \times F^L \times \begin{bmatrix} V_L \\ 0 \end{bmatrix} \qquad (7)$$

where A^i, B, F^S and F^L are 2×2 matrices whose elements are infact arrays of N elements. Here, A^i is a matrix representing the frequency domain response of the ith section of the loop, B is the matrix representing the response of the bridged tap and F^S and F^L are the matrices representing the AFE circuitry for TX (source) and RX (load) paths. From (11) the transfer function of the channel can be derived and is given by

$$H(d_1, d_2, b, f) = \frac{V_L}{V_S}. \tag{8}$$

where d_1 is the length of the section before bridged tap and d_2 is the length of the section after bridged tap. Note that the CO interpretation algorithm uses a two-section, single bridged tap model. This is because of the limited number of frequency bins at which the transfer function is available (from tone 6 to 32 to be exact). With only 26 tones we can't really estimate two bridged taps.

Entries of the above matrices are given as follows:

$$\begin{aligned} A^i_{11} &= A^i_{22} = \cosh(\gamma d_i) \\ A^i_{12} &= Z_0 \sinh(\gamma d_i), \quad A^i_{21} = A^i_{12} Z_0^{-2} \end{aligned}$$

entries of matrix B:

$$\begin{aligned} B_{11} &= B_{22} = 1 \\ B_{12} &= 0, \quad B_{21} = Z_j^{-1}(b) \end{aligned}$$

where Z_j^{-1} is a quantity related to the impedance of the bridged tap and finally:

$$\begin{aligned} F^S_{11} &= F^S_{22} = 1, \quad F^S_{12} = 0, \quad F^S_{21} = Z_S; \\ F^L_{11} &= F^L_{22} = 1, \quad F^L_{12} = 0, \quad F^L_{21} = Z_L^{-1}. \end{aligned}$$

The estimation algorithm tries to minimize the difference between the measured and the actual transfer functions:

$$\min_{d_1, d_2, b} \|H(d_1, d_2, b, f) - Rx(f)\|_2^2. \tag{9}$$

An example of the operation of the US loop length and bridged tap length estimation algorithm is illustrated in Figure 4. Here we display the measured received reverb signal $Rx(f)$ and the theoretical model $H(d, b_1, b_2, f)$ which was obtained by finding the model parameters d, b_1, b_2 that best match the data. The loop consisted of 26 awg 7700 ft wire with a 26 awg 600 ft bridged tap 5900 ft away from CO. Model parameters best matching the observed data were found to be $d_1 = 7900$ ft, $d_2 = 0$ ft and $b = 500$ ft. Note that although d_1 and d_2 parameters found by the algorithm are different than their actual values (actual values are $d_1 = 5900$ ft and $d_2 = 1800$ ft), their sum $d_1 + d_2$ is within 200 ft of the actual loop length. This example shows that even though the loop length is fairly accurate the location of the bridged tap cannot be estimated reliably by this method.

Memory Requirements and Computational Complexity: From the expressions leading to the theoretical channel transfer function, $H(d_1, d_2, b, f)$, it is clear that for the

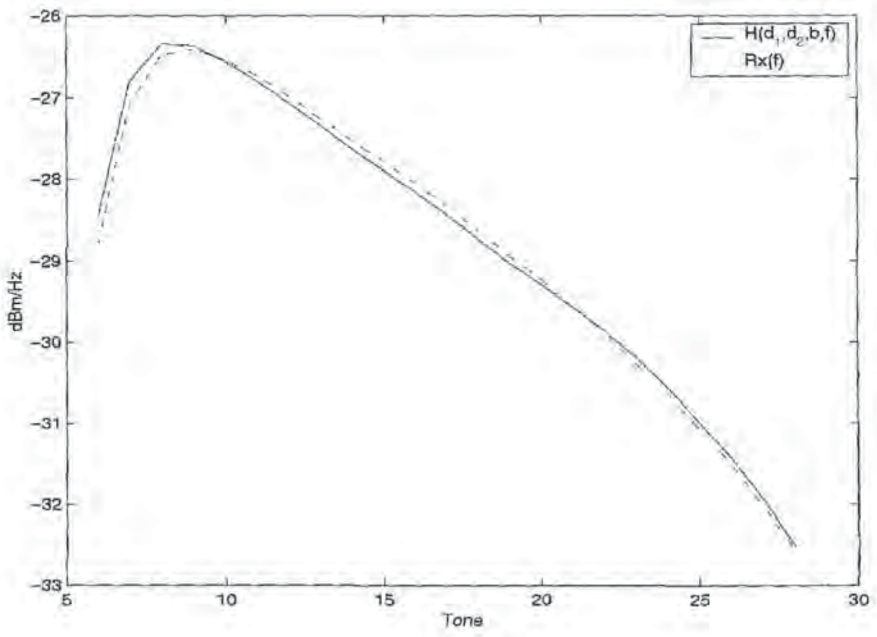

Figure 4: Observed (dashed line) received reverb signal $Rx\,(f)$ versus the theoretical channel model (solid line) $H(d_1, d_2, b, f)$ as functions of frequency for a 7700ft loop with a single 600ft bridged tap.

computation of the theoretical channel response we need to store Z_S, Z_L, Z_0 and γ (for **24awg** and **26awg** only) and $Z_J(b_1)$. Each of these variable are complex arrays of 26 elements. Note that $Z_j(b_1)$ characterizing the bridged tap is dependent on the bridged tap length. Assuming a resolution of 100ft in bridged tap length and a maximum detectable bridged tap length of 2000ft, we have 20 different $Z_J(b_1)$ arrays (this requires 20 locations). Finally we need to store the sinh(.) and cosh(.) elements of the matrices A_1 and A_2. Assuming 500ft resolution in loop length and a maximum measurable loop length of 20,000ft, we need to have 80 $\times$ 46 locations for storing entries of A_i. In total for only storing these variables we need 108 $\times$ 46 memory locations (including storage for $Rx(f)$ and $H(d_1, d_2, b, f)$. Another 10×46 locations are needed for storing intermediate variables during the execution of the algoritm, giving a grand total of approximately 118 $\times$ 46 memory locations.

During the search process, we select P values for d_1, Q values for b and R values for d_2 and compute the MSE for each combination of d_1, d_2, b. To compute the channel impulse response we need 4 $\times$ (8 $\times$ 23complex multiplications + 4 $\times$ 26complex additions). Therefore the total computational cost is $P \times Q \times R \times$ (32 $\times$ 26complex multiplications + 4 $\times$ 26complex additions).

Pseudo-Code:

```
[complex H] = HmodelUS(float d1, float d2, float b, int Gauge, int CODevice,
                    int CPEDevice)
// Inputs:
// d1 = length of the section before bridged tap.
// b1 = length of the section after bridged tap..
// b = bridged tap length.
//
//
// Output:
// H = frequency domain loop model.
{

  gamma = GetGamma( Gauge );      //returns frequency domain propagation function
                                  //for the wire specified by Gauge.
  Z0 = GetZ0( Gauge );            //Returns the frequency domain impedance of
                                  //of the wire specified by gauge.
  Zs = GetZs( CPEDevice );        //Returns the TX impedance of the CPE modem
  Zl = GetZl( CODevice );         //Return the  RX impedance of the CO modem

  //Compute elements of matrix A2.
  for (int i=0; i < N; i++)
  {
      A2_11[i] = cosh( d2 * gamma[i]);
      A2_22[i] = A2_11[i];
      A2_12[i] = Z0[i] * sinh( d2 * gamma[i] );
      A2_21[i] = A2_12[i] / ( Z0[i] * Z0[i] );
  }
  //Compute elements of matric B.
  for (int i=0; i < N; i++)
  {
     B_11[i] = 1.0;
     B_22[i] = 1.0;
     B_12[i] = 0.0;
     B_21[i] = tanh( b * gamma[i] ) / Z0[i];
  }
  //Compute elements of matrix A1.
  for (int i=0; i < N; i++)
  {
      A1_11[i] = cosh( d1 * gamma[i]);
      A1_22[i] = A1_11[i];
      A1_12[i] = Z0[i] * sinh( d1 * gamma[i] );
      A1_21[i] = A1_12[i] / ( Z0[i] * Z0[i] );
  }
  //Compute elements of matrix Fs
  for (int i=0; i < N; i++)
  {
      Fs_11[i] = 1.0;
      Fs_22[i] = 1.0;
      Fs_12[i] = Zs[i];
      Fs_21[i] = 0.0;
  }
  //Compute elements of matrix Fl
  for (int i=0; i < N; i++)
  {
      Fl_11[i] = 1.0;
      Fl_22[i] = 1.0;
      Fl_12[i] = Zs[i];
      Fl_21[i] = 0.0;
  }

  // Multiply the matrices
  // return A = Fs x A1 x B x A2 x Fl
  // A11 is the (1,1) element of the 2x2 matrix A.
  A11 = MatrixMultiply( Fs, A1, B, A2, Fl );

  for (int i=0; i < N; i++)
  {
      H[i] = -10*log10( A11[i] );
  }

  return H;

}


[float dBest, float bBest] = LoopUS( complex Rx[], int Gauge,
                                     int CODevice, int CPEDevice, int N)
// Inputs:
// Rx = calibrated and compensated reverb signal in frequency domain.
// Gauge = reference gauge of the wire.
// CODevice = ID of the CO modem collecting US reverb.
// CPEDevice = ID of the CPE modem transmitting US reverb.
// N = number of elements in Rx[].
//
// Outputs:
// dBest = Estimated loop length.
// bBest = Estimated bridged tap length.
{

 int d1Step = 500, d2step = 500, bStep = 100;

 float MinMSE = 1e20;
 for (d1 = 0; d1 < d1Max; d1 += d1Step)
   for (d2 = d2Max-d1; d2 >= 0; d2 -= d2Step)
     for (b = 0; b < bMax; b += bStep)
```

```
    {
        H = HmodelUS( d1, d2, b, Gauge, CODevice, CPEDevice );
        mse = 0.0;
        for (i=0; i < N; i++)
        {
           mse += ( H[i] - Rx[i] ) * ( H[i] - Rx[i] );
        }

        if (mse < MinMSE)
        {
           mse = MinMSE;
           dBest = d1 + d2;
           bBest = b;
        }
    }

 return dBest, bBest ;
}
```

B.1.3 Loop Characterization by Time Domain Reflectometry (TDR)

The channel characterization algorithms described in preceding sections take advantage of a double-ended diagnostics where both CO and CPE modems are available. If the CPE modem is not installed yet or is not operational we use the time domain reflectometry technique to estimate the physical structure of the line.

The data required by the TDR algorithm is obtained by sending a pre-defined signal to the channel and listening for the echo waveform. The echo obtained this way is analyzed to detect impedance discontinuities casued by the bridged taps, open end of the loop, load coils, etc. During TDR measurements, we will use original 552 KHz sampling frequency (AHeDD) for the receiver. The echo cancellor is going to be running in TDR measurements in order to cancel the near-end echo caused by the AFE circuitry of the linecard. If $x_k(n)$, $n = 1, .., 128$, is the sampled version of the received signal at the kth frame at the output of the echo cancellor, the TDR waveform becomes

$$TDR(n) = \frac{1}{K}\sum_{k=1}^{K} x_k(n). \tag{10}$$

Note that the TDR waveform in (10) is obtained by time-domain averaging. Therefore the FFT in the receive path is going to be turned off during the averaging process.

In theory, any impedance discontinuty in the loop causes a reflection which is observed as a pulse whose location and height can be used to estimate the distance of the impedance discontinuity as well as the type (i.e. whether it is caused by a bridge tap or open end of the loop). If multiple impedance discontinuities are present in the loop, analyzing the time domain waveform of the echo signal becomes very complicated. For this reason, we have chosen a model based approach for TDR estimations. The method basically compares the observed echo with that of a model where we assume the channel consists of three sections seperated by two bridged taps as shown in Figure 5. The objective of the TDR analysis is to estimate d_i, $i = 1, 2, 3$ and b_j, $j = 1, 2$ which provide information about location and the lengths of bridged taps as well as the length of the entire loop.

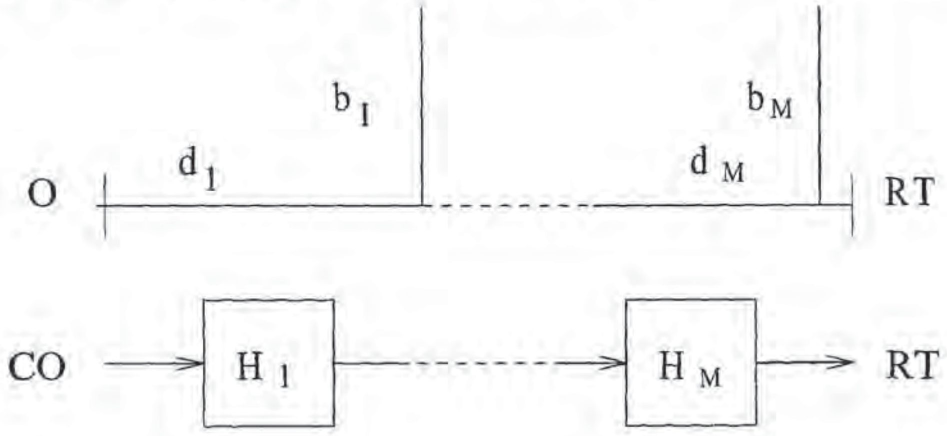

Figure 5: A multiple bridged-tap loop model.

First of all, in the measurement phase we have to make sure that all the phones in the customer premises are *on-hook*. This is necessary since the loop model assumes that the end of the loop is null terminated (open). This requires detection of on/off-hook condition prior to TDR measurements.

Next we perform the TDR measurement by averaging the echo signal over K frames and recording the result. This procedure results in a time domain echo waveform which we will use for comparing with the echo response of a known loop.

The echo response of a loop given d_i and b_j is given by the following equation:

$$\begin{bmatrix} V_S \\ I_S \end{bmatrix} = F^S \times A^1 \times B^1 \times A^2 \times B^2 \times A^3 \times F^L \times \begin{bmatrix} V_L \\ 0 \end{bmatrix} \quad (11)$$

where A^i, B^j, F^S and F^L are 2×2 matrices whose elements are infact arrays of 128 elements. Here, A^i is a matrix representing the frequency domain response of the ith section of the loop, B^j is the matrix representing the response of jth bridged tap and F^S and F^L are the matrices representing the AFE circuitry for TX (source) and RX (load) paths. From (11) the transfer function of the echo path can be derived and is given by

$$H_{echo} = \frac{V_L}{V_S}. \quad (12)$$

Entries of the above matrices are as follows:

$$\begin{aligned} A^i_{11} &= A^i_{22} = \cosh(\gamma d_i) \\ A^i_{12} &= Z_0 \sinh(\gamma d_i), \quad A^i_{21} = A^i_{12} Z_0^{-2} \end{aligned}$$

entries of matrix B^j:

$$\begin{aligned} B^j_{11} &= B^j_{22} = 1 \\ B^j_{12} &= 0, \ \ B^j_{21} = Z_j^{-1} \end{aligned}$$

where Z_j^{-1} is a quantity related to the impedance of the jth bridged tap and finally:

$$\begin{aligned} F^S_{11} &= F^S_{22} = 1, \ \ F^S_{12} = 0, \ \ F^S_{21} = Z_S; \\ F^L_{11} &= F^L_{22} = 1, \ \ F^L_{12} = 0, \ \ F^L_{21} = Z_L^{-1}. \end{aligned}$$

From these equations we can easily compute the required memory size. As already mentioned each of the entries of the matrices given above are actually arrays of 128 complex elements. Since it will be difficult to compute $\cosh(\gamma d)$ and $\sinh(\gamma d)$ values, we need to precompute these quantities in regular intervals, such as 500ft inntervals, from 5kft to 15kft. This requires 42×256 locations for storing cosh(.) and sinh(.) values and 256 locations for storing Z_0^{-2}. Assuming the bridge tap lengths can be distinguished in 250ft steps, we need to allocate 6×256 locations for Z_j^{-1}. We also need 2×256 locations for storing Z_S and Z_L^{-1} values (total so far 52×256 locations). We will also need 8×256 locations for storing intermediate results of multiplications.

The next step is to estimate the complexity of the algorithm. It is clear from our disscussion that the loop length - bridged tap lengths and locations are estimated by minimizing the following with respect to d_1, d_2, d_3 and b_1, b_2:

$$\min_{d_i, b_j} \ |TDR - H_{echo}(d_i, b_j)|^2 . \tag{13}$$

It is clear that we need to search over d_i and b_j parameters. From the location of the first reflected pulse, d_1 and b_1 (if the reflection was caused by a bridge tap) can easily be estimated. So we drop d_1 and b_1 from our search. First three matrices in the expression for echo response, $F^S \times A^1 \times B^1$ can be lumped together and need not be considered. For each set of search parameters, d_2, b_2, d_3, we need to construct the echo response which requires 20×128 complex multiplications and 10×128 complex additions. Then we compute the difference between the actual and theoretical echo responses which requires 128 complex additions. We repeat this procedure as many times as needed by the search algorithm. Since the search algorithm generally needs a variable number of iterations to arrive at the optimal d_2, d_3, b_2 values we can't predict this number.

B.2 Crosstalk/Disturber Estimation

Aside from estimating the elements such as loop length and bridged tap lengths that form the physical structure of a loop, the interpretation algorithm is also able to identify various crosstalk and disturbance sources on the channel. Twisted cable pairs are typically bundled as 25 or 50-pair units. Different DSL services such as HDSL, T1 or ISDN carried by one or more of the twisted pairs are usually picked up by the remaining twisted pairs in the bundle and obserbed as noise sources. The interference entering a twisted pair through some coupling path with other twisted pairs are called crosstalk. There are other sources of disturbance on the line that is caused by electromagnetic coupling. A good example is AM stations. Faulty in-home wiring usually results in the observation of AM signals in DSL frequency band. The objective of the crosstalk/disturber estimation algorithms is to identify the crosstalk sources and provide quantitative information about them such as power level and frequency of the disturber. Identification of a crosstalk/disturber on the line is followed by the rate degradation estimation which is a prediction of the data rate loss caused by the presence of the identified disturber.

From an algorithmic point of view, there are two different algorithms that identify the crosstalk and EMI (ElectroMagnetic Interference). Following two sections describe these two algorithms in detail. Before moving on, we first explain the data collection process.

The interferer detection algorithm uses the power spectrum of the *idle channel noise* (ICN) for the estimation of crosstalk/disturbers on the line. During ICN measurement, we just listen to the channel making sure that there are no meaningful signals, such as an activation request tone, on the line. If we denote the signal present on the receiver as $x(n)$, where $n = 1, .., N$ is the sample index within a frame (N is the number of samples contained in a frame), the power spectrum, $S_{xx}(f)$, is estimated as follows:

$$S_{xx}(f) = \frac{1}{K} \sum_{k=1}^{K} |\mathrm{FFT}_N\left(x_k(n)\right)|^2, \tag{14}$$

where $x_k(n)$ is the sampled signal collected during the kth frame and K is the number of frames that the above averaging is performed. In pain english, we take the N-point signal sequence $x_k(n)$ sampled at the kth frame, take the N-point FFT and average the square of the magnitudes of the FFT coefficients for K consecutive frames. This procedure gives us the periodogram estimate of the power spectrum. As was the case with reverb signal measurement, the power spectrum is available only at a discrete set of frequencies, $f_i = i\Delta f$, $i = i_f, \ldots, i_l$, where i_f and i_l denote first and last tones the power spectrum is sampled at. Accumulation process continues until we obtain desired precision in noise measurement process ($K = 512$ or $K = 1024$ accumulations should provide excellent results). The next step is to estimate interference type present on the channel which is the subject of next two sections.

B.2.1 Crosstalk Estimation Algorithm

The crosstalk type and power are estimated by comparing the measured noise power spectrum to known crosstalk spectral masks such as DSL Next, HDSL Next, T1 Next etc . The exact algorithm proceeds as follows:

1. Minimize with respect to i (denotes ith known disturber), g (power of the disturber) and σ (represents the power of white noise)
2. $MSE_i(g, \sigma) = \sum_{n=6}^{256} \left| PSD_{ICN}(n) - \left(g^2 PSD_i(n) + \sigma^2\right)\right|^2$
3. find the disturber which minimizes the MSE as given above.

In the above algorithm, we vary i, g and σ which are associated with the disturber type, power and white noise level respectively (for example $i = 1$ may denote DSL Next disturber and g denotes its power) and choose the set of variables which minimizes MSE over all candidate crosstalk types. Memory requirements for this algorithm are 256 locations to store ICN power spectra and 256 locations to store the power spectra of each known disturber. If there are P different types of known disturbers for example the storage requirement is $P \times 256$. It must be noted that the storage requirements can be reduced by computing the PSD of the given crosstalk on the fly rather than using 256 locations to store the entire spectrum. That is data memory can be traded off with program memory. We need approximately 350 additional locations for storing intermediate variables during the execution of MSE search algorithm.

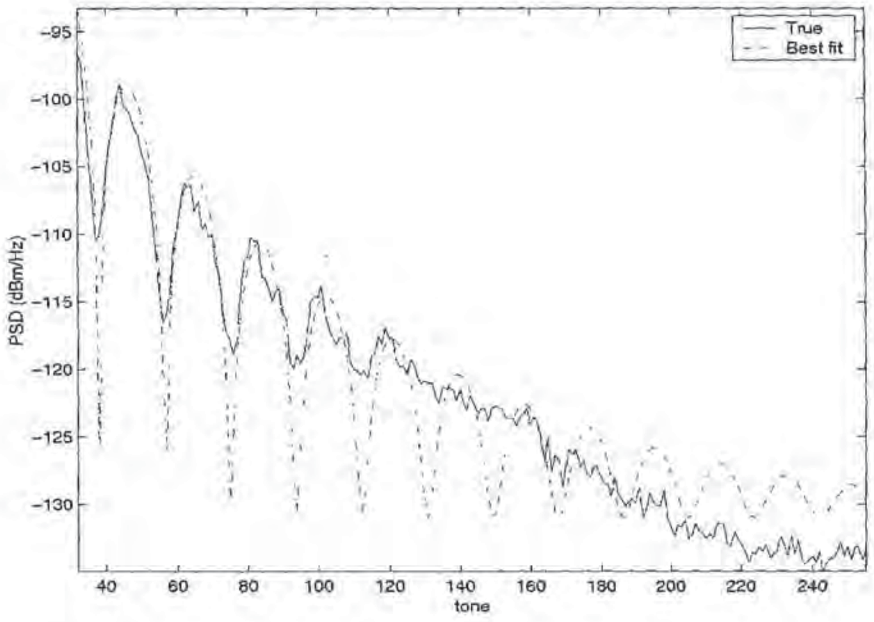

Figure 6: Observed (solid line) PSD of idle channel noise with a DSL Next disturber versus the PSD of the the crosstalk that best fits the true PSD. In this example, interpretation routine determined that the crosstalk type was DSL Next with -35 dBm power.

As for the search algorithm which will be used to determine the parameters i, g and σ which minimizes the MSE, we note that it is easy to detect the background white noise level so it can be dropped from the search algorithm. Then we are left with minimizing MSE with respect to g for each i which can be accomplished easily by picking Q possible values for g and finding, over these Q predetermined values, the one minimizing MSE. In this case total computational cost is: $P \times Q \times$ (500 additions + 250 multiplications). Typical values for P and Q are $P = 5$ (five known disturber PSD's) and $Q = 50$.

An exmple of the operation of the crosstalk detection algorithm is illustrated in Figure 6. Here we display the measured PSD of the ICN (solid line) versus the PSD of the crosstalk that best matches the observed data. The actual disturbance on the line was a DSL Next disturber with -35 dBm power and the crosstalk detection algorithm found exactly the same answer.

Pseudo-Code:

```
[int pBest, float gBest] =  EstimateCrosstalk( float ICN[])
// Input:
// ICN = array containing idle channel noise.
//
// Outputs:
// pBest = index representing the estimated crosstalk type.
// gBest = estimated power level of the crosstalk.
{

 MinMSE = 1e20;
 // Go through each of the P known disturbers
 for (p=0; i < P; p++)
 {
     PSD = GetPSD( p );     //Returns the PSD of the pth known crosstalk
                            //disturber
     //g = power level of the disturber
     for (g = gMin; g < gMax; g += gStep)
     {
         mse = 0.0;
         for (i=0; i < N; i++)
         {
             mse += ( ICN[i] - (  g^2 * PSD[i] + sigma^2 ) )^2;
         }

         if ( mse < MinMSE)
         {
            mse = MinMSE;
            gBest = g;
            pBest = p;
         }
     }
 }

 return pBest, gBest;

}
```

B.2.2 AM/EMI Detection Algorithm

An ADSL receiver is susceptible to AM/EMI interference because part of the ADSL receive band coincides with AM and amateur radio broadcast frequencies. According to FCC specifications, AM radio broadcast frequencies starts at 540 KHz and extends upto 1.8 MHz. Beyond this frequency band it is possible to find EMI ingress caused by the amateur radio broadcast from 1.9 MHz to approximately 3.3 MHz. In home wiring connecting the ADSL modem to the telephone line usually acts as a huge antenna picking up the AM/EMI sources.

Figure 7 (a) shows the power spectrum of a typical AM/EMI interference pattern with multiple AM interferers. The data was collected by running the data collection piece on an ADI 918 CPE. AM broadcast is done by modulating a baseband signal such as voice or music by "amplitude modulation". Denoting the baseband signal by $f(t)$, t being time, the modulated signal is given by

$$e_m(t) = f(t)\cos(w_c t) + A\cos(w_c t) \tag{15}$$

where A is a constant and $w_c = 2\pi f_c$ is the radian carrier frequency. From (15) it is obvious that the spectrum of $e_m(t)$ consists of the baseband signal shifted in frequency by $\pm w_c$ plus two additional pulses at $\pm w_c$. Therefore

$$\text{FFT}\left(e_m(t)\right) = \frac{1}{2}\left[F(w - w_c) + F(w + w_c)\right] + \pi A\left[\delta(w - w_c) + \delta(w + w_c)\right] \tag{16}$$

The AM/EMI interference detection problem is complicated by the fact that the observed spectrum is dependent on the unknown spectrum of the time-varying baseband signal $f(t)$ as shown in (16). It is clear that any AM/EMI detection algorithm should use only the carrier frequency of the modulating wave as a signature. We may estimate the AM/EMI interference frequency and power by modeling the power spectrum of AM/EMI as a constant background noise plus a number of spikes, parameterized by the frequency and height, representing the AM/EMI carrier frequencies. Then we compare our model with the observed spectrum by varying the frequency and height of each individual spike. The frequency/height configuration of the model best matching the original power spectrum in terms of mean square error is declared as our estimation. However, such an approach has one serious drawback. Since each spike is parameterized by 2 parameters (frequency/height), each additional AM/EMI disturber adds 2 more parameters to our optimization problem. If there are ten AM/EMI disturbers for example, we have to carry out our optimization over 20 parameters, which is too complicated. Fortunately, there is an easier way. Looking at the spectrum of the AM/EMI interferers, we realize that the first derivative of the spectrum at carrier frequencies is not continuous. That is at the carrier frequency, the slope of the spectrum jumps abruptly from a positive large number to a negative large number (This fact can be proven mathematically). This means that the second derivative of the spectrum contains large negative pulses and these can easily be detected by setting a negative threshold and finding those impulses whose heights are below the set threshold. Figure 7 shows the described AM/EMI detection algorithm in action. Figure 7 (a) shows the power spectrum of the ADSL receive band which contains a number of AM/EMI disturbers. Figure 7 (b) shows the second difference (corresponds to second derivative in continuous time) of the power spectrum in (a). We observe large,

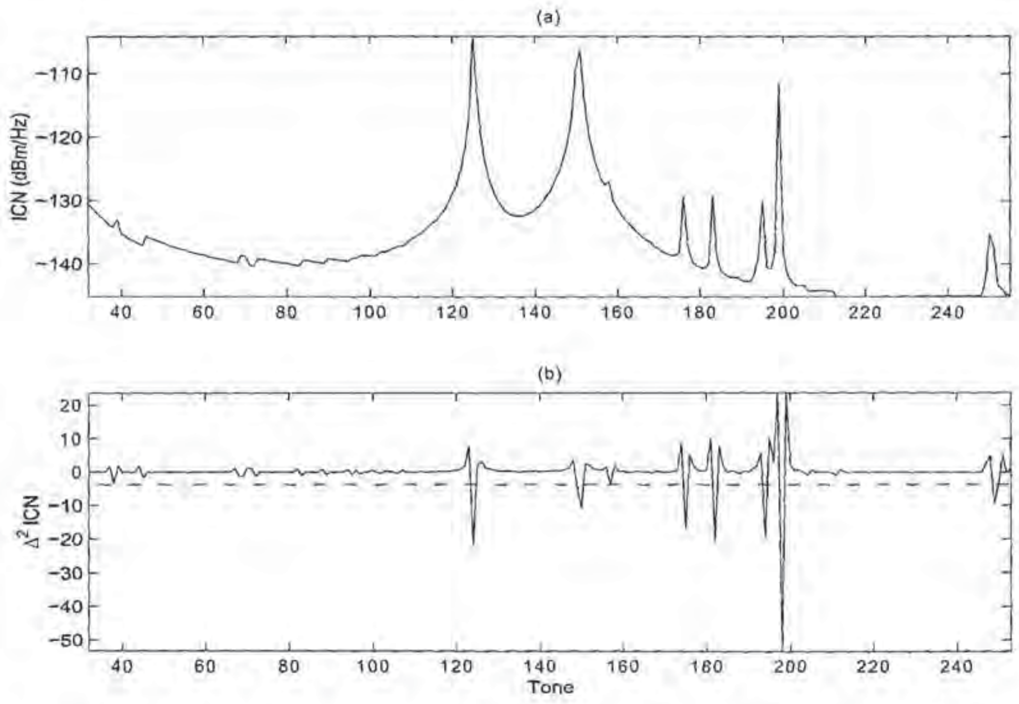

Figure 7: (a) Observed power spectrum of AM/EMI interference pattern (b) Second difference of the power spectrum in (a). Dash-dotted line denotes the threshold below which the presence of an AM/EMI interference is declared.

negative spikes at the points where AM/EMI carrier frequencies are located. The carrier frequencies are detected by simply finding points in Figure 7 (b) where the second differences exceed a set threshold (shown by dash-dotted line in Fig. 7 (b)). The power of each AM/RFI disturber is then estimated directly from the original power spectrum.

Pseudo-Code:

```
[int AMTone[], float AMPower[], int NoOfAM] =  EstimateAM(float ICN[])
// Input:
// ICN = array containing idle channel noise.
//
// Outputs:
// AMTone = array containing the tone numbers corresponding
//          to the detected AM/EMI disturbers.
// AMPower = array containing the power level of the
//           AM/EMI disturbers.
// NoOfAM = number of AM disturbers.
{

 //Compute second difference of the ICN
 for (i = 1; i < N-1 ; i++)
 {
     d2ICN[i-1] = ICN[i-1] - 2.0 * ICN[i] + ICN[i+1];
 }

 NoOfAM = 0;
 for (i=0; i < N-1; i++)
 {
     if ( d2ICN[i] < Threshold )
     {
        AMTone[ NoOfAM ] = i + 1;
        AMPower[ NoOfAM ] = ICN[i+1];
        ++NoOfAM;
     }
 }

 return AMTone, AMPower, NoOfAM;
}
```

B.3 Rate Degradation Estimation

One of the functions of the interpretation software is the estimation of rate reduction caused by the presence of crosstalk/disturbers on the line. If the crosstalk/disturber detection algorithm determines that there are noise sources other than background white noise on the line, the algorithm updates the available SNR tables (obtained either by single-ended or double-ended diagnostics) by undoing the SNR reduction caused by the disturbers. It then runs the bitloading routine on the updated SNR table with given margin, framing and coding info to get the rates for a disturber-free line. The difference between the actual and estimated data rates gives us the rate reduction caused by the noise sources. In order to formulate the problem at hand we note that SNR is given by

$$SNR(f_i) = 10\log_{10}\frac{|H(f_i)|^2}{S_{xx}(f_i)} \tag{17}$$

where $H(f_i)$ is the channel impulse response evaluated at the ith tone and $S_{xx}(f_i)$ is the PSD of the noise on the line evaluated at the ith tone. If there were no noise sources on the line except for the background white noise, the expression for SNR would simplify to

$$SNR_{No-Disturber}(f_i) = 10\log_{10}\frac{|H(f_i)|^2}{\sigma^2} \tag{18}$$

where σ is the standard deviation of the white noise. It is easy to see from equations (17) and (18) that once a disturber is detected we can easily compute $SNR_{No-Disturber}$ given $S_{xx}(f_i)$, the actual PSD of the noise (ICN), and σ. The next step is running the bitloading routine on $SNR_{No-Disturber}$ and computing the rate difference corresponding to SNR and $SNR_{No-Disturber}$.

The rate degradation algorithm uses 26 temporary memory locations for US and 224 temporary locations for DS to hold $SNR_{No-Disturber}$. The algorithm uses existing bitloading routine for computing estimated data rate for a disturber-free line. Hence the program memory required to implement the algorithm is very low.

Pseudo-Code:

```
[int RateReduction] = EstimateRateReduction( float ICN[], float ICNNominal[],
                                             float SNR[], float Margin,
                                             int FramingMode, float CodingGain,
                                             int N)
// Input:
// ICN = array containing idle channel noise in dB.
// ICNNominal = array containing ICN with no crsstalk or AM/EMI disturbers (in dB).
// SNR = SNR Medley.
// Margin =  margin.
// FramingMode = framing mode used in training.
// CodingGain = gain of the code used.
// N = number of elements in the SNR table.
//
// Outputs:
// RateReduction = Estimated rate reduction caused by the disturbers on the line.
{

 // Compute the data rate from the SNR table.
 RateWithDisturbers = Bitload( SNR, Margin, FramingMode, CodingGain );

 //Undo the SNR reduction caused by disturber
 for (i = 0; i < N ; i++)
 {
     SNRNoDisturber[i] += ICN[i] - ICNNominal[i];
 }

 //Compute the estimated maximum data rate
 RateWithoutDisturbers = Bitload( SNRNoDisturber, Margin, FramingMode, CodingGain );
 RateReduction = RateWithoutDisturbers - RateWithDisturbers;

 return RateReduction;

}
```

B.4 Data Rate Estimation

As mentioned previously, TDR data is used to estimate loop length and bridged tap lengths. From the information extracted from the TDR interpretation algorithm one can estimate the frequency domain channel impulse response, $H(f_i)$. We also have the PSD of the noise, $S_{xx}(f_i)$, from the ICN measurement. SNR cane be estimated from these two quantities and is given by

$$SNR(f_i) = 10\log_{10}\frac{|H(f_i)|^2}{S_{xx}(f_i)}, \quad i = i_s, \ldots, i_l \tag{19}$$

where i_s and i_l are the first and last tones $SNR(f_i)$ is evaluated. The data rate is estimated by running the bitloading routine on the estimated SNR with given margin, framing and coding info. Since the rate estimation algorithm uses existing bitloading routines it requires very little data memory. A scratch memory of 26 locations US and 224 locations for DS is required to store estimated SNR.

Pseudo-Code:

```
[int DataRate] =  EstimateDataRate( float H[], float ICN[], float Margin,
                      int FramingMode, float CodingGain, int N)
// Input:
// H = estimated channel attenuation in dB.
// ICN = idle channel noise in dB.
// Margin =  margin.
// FramingMode = framing mode used in training.
// CodingGain = gain of the code used.
//
// Outputs:
// DataRate = Estimated data rate.
{


 // Compute SNR table.
 for (i = 0; i < N ; i++)
 {
     SNR[i] = H[i] - ICN[i];
 }
 //Perform bitloading on SNR and compute the estimated  data rate
 DataRate = Bitload( SNR, Margin, FramingMode, CodingGain );

 return DataRate;

}
```